APPLICATION

FOR

UNITED STATES LETTERS PATENT

APPLICANT NAMES: Karen Appleby
Liana Liyow Fong
German Sergio Goldszmidt
Srirama Mandyam Krishnakumar
Donald Philip Pazel

TITLE: METHOD, SYSTEM, AND PROGRAM PRODUCTS FOR DISTRIBUTED CONTENT THROTTLING IN A COMPUTING ENVIRONMENT

DOCKET NO.: YOR920010330US1

INTERNATIONAL BUSINESS MACHINES CORPORATION

CERTIFICATE OF MAILING UNDER 37 CFR 1.10
I hereby certify that, on the date shown below, this correspondence is being deposited with the United States Postal Service in an envelope addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231 as "Express Mail Post Office to Addressee"
Mailing Label No. _____
on _____

_____
Name of person mailing paper

_____ _____
Signature Date

METHOD, SYSTEM, AND PROGRAM PRODUCTS FOR DISTRIBUTED CONTENT THROTTLING IN A COMPUTING ENVIRONMENT

Field of the Invention

The present invention generally relates to managing load on specific components in a web server farm. More specifically, it allocates the Internet load for web components such as application components using Enterprise Java Beans® (EJB), databases, servlets, etc., in a server farm by controlling the admittance of requests based on the content of the request. Additionally, the method uses the distributed processing capability inherent in the web serving environment to limit the impact on overall system performance.

Description of the Related Art

The main function of a web server farm is to serve HTML (Hyper Text Markup Language) web pages to clients using web browsers such as Netscape® or Internet Explorer®. The HTML pages are generated at a component known as the web server. The web server generates the pages by processing the HTTP (Hyper Text Transfer Protocol) requests made by the web browser clients. The HTTP request includes the web page identifier in the form of a URL (uniform request locator) and possibly several other parameters such as cookies that could be used by the web server. The web server processes the requests by either fetching the web pages from a local repository that could be on a hard drive or by

passing the request to an external component that may generate the web page. Examples of these external components are application servers, CGI (Common Gateway Interface) scripts etc. These components may be located on a different hardware platform from the web server.

An example of this type of request would be a request to display the cost items in a shopping cart. The external module in this case may interact with a backend database server to obtain the cost of the items. It then creates a HTML web page customized for this request and sends the page back to the web server. A web server farm includes a heterogeneous mix of components such as web servers, application servers, database servers, etc., that perform different functions. These components may run on different types of hardware platforms, typically just referred to as servers, with different capabilities such as amount of memory, processor speed, and type.

The servers are physically interconnected by networks which are in turn connected to the Internet. Figure 1 shows an example of a conventional web server farm. The servers are logically connected in a tiered manner such that one or more web servers 30 forming the outermost tier are followed by the application servers 40 and database servers 50.

The web browsers make a TCP connection request to the web server via the Internet 10 and send the HTTP request on this connection. Some networks could have IP L4LB (Layer 4 Load Balancing) components 20 which distribute the

incoming TCP connections from the web browsers to the web servers that are on the network. When the web browser makes a TCP connection request, the L4LB 20 redirects the request to a web server 30 that is capable of handling the request. The web server then processes this request as explained above.

One problem with IP layer 4 based load balancing is that it does not look at the payload of the data and there is no differentiation among requests. That is, there is no differentiation between a request for a static web page that is stored in a repository such as a disk and a request to create a dynamic page by obtaining information from a backend database. This leads to the following problems when connection requests are subjected to admission control at the dispatcher level.

First, requests for static pages that are relatively small and impose very little load on the system could get dropped. This is related to the way in which the IP L4 load balancers work. The L4LB's use the TCP header that initiates a connection to determine admission. A request to fetch a web page is sent on a TCP connection. Furthermore, there could be many HTTP requests that could be sent on a TCP connection.

Second, this method could cause web site outages as the requests to overloaded components do not get sufficiently throttled, thereby causing them to crash. For purposes of the present Application, "throttling" means a method of dropping requests when the load on the processing component increases.

A web server farm includes a finite number of computing resources that are shared for processing requests from web clients. Different clients need different functions such as obtaining a static web page or obtaining a list of items in a catalog that is typically handled by a servlet or an EJB. These functions impose different loads on the various computing resources and the functions have varying levels of importance for the business that is hosting the web site.

The current state of the art treats all the requests to be of equal priority and processes the requests in the order in which they arrive. If there is an overload on the overall system, then requests are simply dropped by the L4LB. However, businesses that are conducted over the Internet suffer from this approach as there is no priority-based dropping of the request. For example, a customer who is requesting a web page to submit his credit number for processing is clearly more important than a customer requesting to view the picture of an item in the catalog.

SUMMARY OF THE INVENTION

The problem with the L4LB's discussed above is that they reject or accept connection requests as opposed to HTTP requests. An analogy that could be used here is the following.

Assume that there is a boat that can sustain a certain weight. When the boat is getting to be nearly full, then it makes sense to admit only light weight people instead of closing the entry to the boat itself. The TCP connection can be thought of as the entry into the boat. The subject of the present invention (e.g., content-based throttling) can be thought of as a "smart" gate keeper who lets people through based on the weight of the people.

In view of the foregoing and other problems, an object of the present invention is to provide a structure and method for distributed throttling which improves the efficiency of a computer network, especially in web hosting server farms.

It is another object of the present invention to provide a structure and method to improve throughput of backend servers due to improved load management.

It is another object of the present invention to prevent crashes of Internet server farms and to provide feedback to a requester that a current request has been rejected.

In order to achieve the above objects and goals, according to a first aspect of the present invention, herein is described a method in a computer network of controlling the admittance of requests to at least one processing component, including differentiating the type of the requests based on the content in each

request and admitting the request only if the differentiated type meets at least one criterion for admission.

According to another aspect of the invention, herein is described a method of controlling the admittance of requests to at least one processing component in a distributed heterogeneous computing environment, each request including a direction component and a message component, the method including receiving a request, evaluating at least a part of the message component of the received request, and providing an admission of the received request based on this evaluation.

According to another aspect of the present invention, herein is described a request throttler in a computer network that controls an admittance of requests to at least one processing component, where the request throttler includes a differentiator to evaluate the message content of each incoming request and a switch to admit each request only if the evaluation passes at least one criterion for admission.

According to another aspect of the present invention, herein is described a computer-readable medium containing computer-readable instructions for a request throttler in a computer network that controls an admittance of requests to at least one processing component, where the request throttler includes a differentiator to evaluate the message content of each of the requests and a switch

to admit each request only if the evaluation passes at least one criterion for admission.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

Figure 1 illustrates an example of the logical configuration of a conventional web server farm;

Figure 2 illustrates a typical exemplary conventional web server farm;

Figure 3 illustrates the concept of throttling in a web server;

Figure 4 is a flowchart illustrating one aspect of a preferred embodiment of content-based throttling of the present invention;

Figure 5 is a flowchart illustrating a second aspect of a preferred embodiment of a method for content-based throttling of the present invention;

Figure 6 is a flowchart illustrating an exemplary decision process of the content-based throttling of the present invention;

Figure 7 illustrates an exemplary hardware/information handling system 700 for incorporating the present invention therein; and

Figure 8 illustrates a signal bearing medium 800 (e.g., storage medium) for storing steps of a program of a method according to the present invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT OF THE INVENTION

The present invention relates to managing load on specific components in a web server farm. As described above, a web server farm includes a heterogeneous mix of components with different capabilities and perform different functions such as a web serving, database serving, application serving, etc. These components are typically structured in a tiered manner so that requests flow from one tier to the other.

Figure 2 shows an exemplary conventional web server farm. Requests for performing tasks typically arrive into the farm from the Internet 101. The Internet 101 forwards the request into the router 102, which is typically located at the entry to the farm or at a location in the Internet Service Provider's point of presence. The router uses the IP address contained in the incoming packet to route the request to one of the L4LBs 103,104. The L4LB provides a virtualization of the IP addresses of all the servers that are connected to it. The L4LB distributes the TCP connection requests using weights assigned to the web servers 105-112 that are connected to it. This distribution is typically done during the arrival of a TCP connection and not during each individual HTTP 1.1 request made on the connection.

In this typical configuration, servlets 113,115 might process certain of the requests. These servlets are typically written in Java® and run in a Java Virtual Machine container such as a Websphere application server, JServ or TomCat. Certain types of requests might be processed by CGI bin application 114. Application server 116 might contain components such as Enterprise Java Beans used to perform business logic functions. Database 117 contains data that is critical to the processing of requests served by the other components such as the servlet or application server.

Figure 3 depicts one exemplary embodiment of the present invention by showing a possible organization of components in a web serving environment that supports content based throttling. Web server 201 serves as a request processor at which the requests arrive after they have been dispatched by the L4LB 103, 104. Thus, web server 201 might correspond to a web server 105-112 shown in Figure 2. In this exemplary embodiment, the web server request processor 201, content throttler 202, and content handler 203 are all implemented as part of the web server 105-112. Because of this repetitive installation throughout the web server farm, the invention additionally can have a distributive aspect.

Web servers 105-112 typically provide a feature where one can plug in software components that are typically not part of the original web server. These components can be interposed in the request processing phases of the web server to provide additional functionality or modify the functionality provided by the base

web server. Such software modules are referred to as "plugins" and are common in the art. The content throttler 202 can be implemented as a plugin in this embodiment. While the content handler 202 could be typically a part of the core web server, it can be replaced by a plugin that can process specific types of requests. An example of such a scenario would be a function that replaces a file handler for a streaming video image with a custom page handler only when the bandwidth utilization is high.

The request processing in a web server 105-112 typically includes inspecting the URL of a received request and determining the content handler that can process the request based on the type of the request. In contrast, the conventional method has only the following steps in a web server: determine the type of request based only the URL, and, select the content handler that would handle the request (the web server configuration provides the information required for mapping the request type to the content handler).

Thus, the method of the present invention interposes an additional step that determines whether or not the request needs to be sent to the content handler based on the "content" of the request (i.e., the complete payload of the request such as information in the POST request). A request type is identified by the configuration information that is present in the web server configuration. Requests can be broadly classified into static and dynamic requests. Static requests are

requests for information that are contained in a static repository such as HTML file or a GIF image file.

Dynamic requests are for information that has to be constructed as a response to the request. Dynamic requests cannot be simply stored in a repository and presented to the user when the request has been made. Typical examples of dynamic requests are for information such as items in the shopping cart of a client or information about the stock portfolio of the client.

The web server configuration contains information that maps the type of request with the content handler that can process the request. Currently, the only web server configuration that maps the type of request to a content handler is based only on the URL. The content handler then processes the request by performing the necessary tasks to deliver the content requested. If the request is for a static HTML file or a GIF image it obtains the file by making an operating system dependent call to get the file and delivers the file on the connection that has been established between the web server and the client's web browser. If the request is for obtaining the items in the shopping cart of a client, then the content handler may initiate a query to a backend database to fetch the items in the catalog, construct a HTML web page with the result of the query after formatting the web page suitably, and deliver the page to the web browser client on the connection that has been established between the browser and the server.

Figure 4 depicts a process 300 of content based throttling in a web serving environment according to a preferred embodiment of the invention. At step 301, a request arrives at the web server request processor 201 (Figure 3) after a connection has been established.

At step 302, the web server 105-112 (Figure 2) determines if the request should be subjected to content based throttling. This process is explained further in the following sections and in Figure 5. A key factor in this determination is the current activity of the web farm. For example, if traffic is light, then there is obviously no need for throttling.

If it is determined that the request does not need to be throttled, then the request is simply handed over to the content handler, and at step 304, the content handler 203 (Figure 3) produces the content, such as a HTML page that contains a description of the items in a shopping catalog. In this example, database 204 (Figure 3) storing the shopping catalog information would be consulted by content handler 203 (Figure 3) to create the HTML page. In step 306, the response is sent back to the web server.

If the request needs to be throttled due to high current traffic or other criterion used for the throttling threshold decision, then the web server hands over the request to the content throttler 202 (Figure 3). In step 303 (Figure 4), the content throttler 202 determines whether a request should be processed, based on one or more or a combination of such metrics such as load on the content handler,

load on the database, load on the server that is running the application components, or available bandwidth. Other metrics might be appropriate for specific applications and would be apparent to one skilled in the art. A simple scenario illustrating the concept is discussed below relative to Figure 6.

If the content throttler determines that the request should not be processed, then it sends back a custom page in step 305 (Figure 4) to the web server. This page could contain information that describes that the web site is experiencing peak load and the user at the browser could try a while later.

If the content throttler determines that the request could be admitted, then it hands the request to the content handler to process the request in step 304. Either way, the web server obtains the content that is obtained by the content handler (step 306) or the content throttler (step 305) and sends the response back to the browser that originated the request.

Figure 5 provides a simple exemplary technique 400 of the step that determines if a request needs to be subjected to content based throttling. As mentioned above, in a preferred embodiment of the invention, this determination is relevant only if traffic on the farm has exceeded a preselected threshold. In step 401, the web server determines the type of the request based on the URL in the request and the web server configuration. There is a mapping in the web server configuration that contains the plugins that perform additional processing on request of certain types.

The web server checks in step 402 if the request type is flagged to be processed by the content-based throttler. If the flag is set, then this request needs to be subjected to content based throttling (step 403). If the flag is not set, then there is no need to subject the request to content based throttling (step 404).

Figure 6 depicts an exemplary decision process 500 for content based throttling, as done by content throttler 202 (Figure 3). The request is identified by making use of the configuration information in the web server configuration and matching it against the load parameter that is monitored. In the example shown in Figure 6, the request is identified as being for a backend database.

As shown in this example, in step 502, the request is associated with the response time as the load parameter. There is a threshold for the response time that has been set either at configuration time or dynamically by a component that is responsible for determining the load on the database.

In step 503, the content throttler determines whether the request should be admitted based on the current value of the response time, a measurement continuously being done by the system as a measurement of current activity for the backend database. If the current response time is less than the threshold that has been set then the request is admitted (step 504). Otherwise, the request is dropped (step 505).

The alternative route shown in step 502 refers to a "request for CPU bound CGI script". As explanation of this alternative, the web server configuration

contains request types. This request type (CGI) can be considered as a coarse grained request. The request type classification used by the content throttler can be considered a finer grained classification within a single request type identified by the web server configuration. The "request for CPU bound CGI script" depicts this classification. This is only one illustration of a process of classification of the requests.

Another example of a content throttler might be based on the size of the output being generated for requests to images stored in a repository. The throttler obtains the size of the image from the repository and, if the size of the image is less than a threshold size, then it may decide to admit the request. Otherwise, it may choose to send a custom HTML page refusing the admission of request. Note that these are only several of the algorithms possible for the throttling decision. Other algorithms could be used to arrive at the throttling decision, as would be apparent to one skilled in the art, taking the present disclosure as a whole.

The embodiment described above is distributed in nature, as the throttling is performed at every web server in the system. This approach is different from the conventional systems and methods where the admittance control decisions are made at the level of L4LB which is a central entity. Content based routing (CBR) which is sometimes implemented on top of L4LB routes requests to web servers based on the content. Implementations of CBR make use of a set of rules to match

URL's with the rules and tend to be bottlenecks as they inspect every request that arrives to the web site.

Figure 7 illustrates a typical hardware configuration of an information handling/computer system in accordance with the invention and which preferably has at least one processor or central processing unit (CPU) 711.

The CPUs 711 are interconnected via a system bus 712 to a random access memory (RAM) 714, read-only memory (ROM) 716, input/output (I/O) adapter 718 (for connecting peripheral devices such as disk units 721 and tape drives 740 to the bus 712), user interface adapter 722 (for connecting a keyboard 724, mouse 726, speaker 728, microphone 732, and/or other user interface device to the bus 712), a communication adapter 734 for connecting an information handling system to a data processing network, the Internet, an Intranet, a personal area network (PAN), etc., and a display adapter 736 for connecting the bus 712 to a display device 738 and/or printer 739 (e.g., a digital printer or the like).

In addition to the hardware/software environment described above, a different aspect of the invention includes a computer-implemented method for performing the above method. As an example, this method may be implemented in the particular environment discussed above.

Such a method may be implemented, for example, by operating a computer, as embodied by a digital data processing apparatus, to execute a

sequence of machine-readable instructions. These instructions may reside in various types of signal-bearing media.

Thus, this aspect of the present invention is directed to a programmed product, comprising signal-bearing media tangibly embodying a program of machine-readable instructions executable by a digital data processor incorporating the CPU 711 and hardware above, to perform the method of the invention.

This signal-bearing media may include, for example, a RAM contained within the CPU 711, as represented by the fast-access storage for example. Alternatively, the instructions may be contained in another signal-bearing media, such as a magnetic data storage diskette 800 (Figure 8), directly or indirectly accessible by the CPU 711.

Whether contained in the diskette 800, the computer/CPU 711, or elsewhere, the instructions may be stored on a variety of machine-readable data storage media, such as DASD storage (e.g., a conventional "hard drive" or a RAID array), magnetic tape, electronic read-only memory (e.g., ROM, EPROM, or EEPROM), an optical storage device (e.g. CD-ROM, WORM, DVD, digital optical tape, etc.), paper "punch" cards, or other suitable signal-bearing media including transmission media such as digital and analog and communication links and wireless. In an illustrative embodiment of the invention, the machine-readable instructions may comprise software object code, compiled from a language such as "C", etc.

The invention as described above provides a number of benefits to web servers. It provides overload protection of backend servers. Its distributed aspect of throttling improves the efficiency of the system and its improved load management provides possible improvement in throughput of backend servers. The invention is particularly beneficial to web hosting server farms.

While the invention has been described in terms of a single preferred embodiment, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims.